

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-272717

(43)Date of publication of application : 18.10.1996

(51)Int.Cl.

G06F 13/00  
G06F 13/10

(21)Application number : 07-324647

(71)Applicant : MICROSOFT CORP

(22)Date of filing : 13.12.1995

(72)Inventor : WILLIAMS ROBERT J  
GAPUTO CHRISTOPHER P  
LAEPPLER KEITH A

(30)Priority

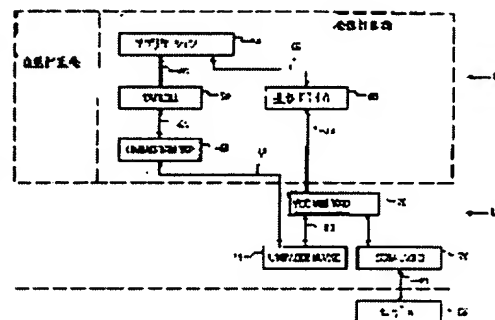
Priority number : 94 356059 Priority date : 13.12.1994 Priority country : US

## (54) MODEM INTERFACE

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a modem interface which does not depend on a device.

SOLUTION: This interface uses call control for an application program and a data transmission application program interface and accesses a modem by a method which does not depend on a device. A general purpose modem driver 74 reads device specified information from a registry and generates a specified control command to a modem 56. The driver 74 divides modem transparency for the application 54 into command and data modes. This modem interface detects the modem 56 and contains support which loads the device specified information to a registry based on modem information.



## LEGAL STATUS

[Date of request for examination] 13.12.2002

[Date of sending the examiner's decision of rejection] 05.09.2006

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-272717

(43) 公開日 平成8年(1996)10月18日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 13/00	3 5 3	7368-5E	G 0 6 F 13/00	3 5 3 C
13/10	3 3 0	7368-5E	13/10	3 3 0 C

審査請求 未請求 請求項の数19 O L (全 16 頁)

(21) 出願番号 特願平7-324647

(22) 出願日 平成7年(1995)12月13日

(31) 優先権主張番号 08/356059

(32) 優先日 1994年12月13日

(33) 優先権主張国 米国 (U S)

(71) 出願人 594115430

マイクロソフト コーポレーション  
Microsoft Corporation  
アメリカ合衆国 98053-6399 ワシントン州 レドモンド ロケーション 8エス-2076 ワン マイクロソフト ウェイ  
(番地なし)

(74) 代理人 弁理士 杉村 暁秀 (外4名)

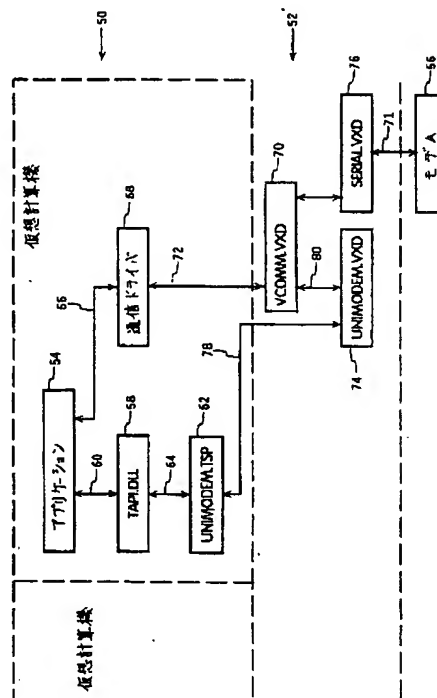
最終頁に続く

(54) 【発明の名称】 モデムインタフェース

(57) 【要約】

【課題】 デバイスに依存しないモデムインタフェースを提供する。

【解決方法】 本インタフェースは、アプリケーションプログラム用の呼び制御およびデータ伝送アプリケーションプログラミングインタフェースを用い、デバイスに依存しない方法でモデムにアクセスする。汎用モデムドライバは、デバイス特定情報をレジストリから読み取り、特定制御コマンドをモデムに対して発生する。汎用モデムドライバは、アプリケーションに対するモデム透過性をコマンドモードとデータモードとに分離する。本モデムインタフェースは、モデムを検出し、デバイス特定情報をモデム情報に基づくレジストリにロードするサポートを含む。



## 【特許請求の範囲】

【請求項1】 1つまたはそれ以上のアプリケーションプログラムに対するデバイスに依存しないモデムインタフェースにおいて、

モデム特定制御情報を格納するレジストリデータ記憶装置と、

モデムに依存しない呼制御コマンドを受け、モデムを制御する呼コントローラと、

モデムに依存しないデータI/Oコマンドを受け、モデムを経てデータを転送する通信ドライバとを具え、

汎用モデムドライバが前記レジストリデータ記憶装置と通信してモデムに関するモデム特定制御情報を読み取り、前記汎用モデムドライバが前記呼コントローラと通信してモデムに依存しない呼制御コマンドを受けるとともに前記通信ドライバと通信してモデムに依存しないデータI/Oコマンドを受け、前記汎用モデムドライバが前記モデムと通信して前記モデムに依存しない呼制御コマンドを前記レジストリデータ記憶装置における前記モデム特定制御情報に基づくモデム特定制御コマンドに変換することを特徴とするデバイスに依存しないモデムインタフェース。

【請求項2】 請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、前記汎用モデムドライバが、モデムのコマンドモードとデータモードとを区別し、モデムがコマンドモードの場合、前記汎用モデムドライバは、前記呼コントローラから呼制御コマンドをモデムに転送し、モデムがデータモードの場合、前記汎用モデムコントローラは、データI/Oコマンドをモデムに転送するように構成したことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項3】 請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、前記レジストリデータ記憶装置が、情報ファイルからモデム特定制御情報をこのレジストリにロードするクラスインストーラと通信することを特徴とするデバイスに依存しないモデムインタフェース。

【請求項4】 1つまたはそれ以上のポートドライバとデータ通信を行うポートインタフェースをさらに具える請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、

前記汎用モデムドライバが、前記1つまたはそれ以上のポートドライバの1つとポートドライバの形式に依存しないで通信するポートドライバインタフェースと通信するように構成したデバイスに依存しないモデムインタフェース。

【請求項5】 請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラおよび通信ドライバを一つのモジュールに結合し、1つまたはそれ以上のアプリケーションプログラムにアクセス可能なアプリケーションプログラミングインタフェース

を設けることを特徴とするデバイスに依存しないモデムインタフェース。

【請求項6】 請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラおよび通信ドライバを別個のモジュールとし、各々が1つまたはそれ以上のアプリケーションプログラムにアクセス可能なアプリケーションプログラミングインタフェースを設けることを特徴とするデバイスに依存しないモデムインタフェース。

10 【請求項7】 請求項6に記載のデバイスに依存しないモデムインタフェースにおいて、前記呼コントローラを動的リンクライブラリとしたことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項8】 請求項6に記載のデバイスに依存しないモデムインタフェースにおいて、前記通信ドライバを動的リンクライブラリとしたことを特徴とするデバイスに依存しないモデムインタフェース。

20 【請求項9】 請求項1に記載のデバイスに依存しないモデムインタフェースにおいて、モデム検出コマンドをモデムに発生し、モデム特定識別情報をモデムから受け、前記モデム特定識別情報に基づくデバイスIDを発生するクラスインストーラをさらに含むことを特徴とするデバイスに依存しないモデムインタフェース。

【請求項10】 1つまたはそれ以上のアプリケーションプログラムにアクセスできる呼コントローラおよび通信ドライバと、モデム特定コマンドをモデムに転送する汎用モデムドライバと、モデム特定制御情報を格納するレジストリデータ記憶装置とを有するコンピュータシステムにおいて、モデムを経て通信する方法において、

30 前記汎用モデムドライバがモデム特定制御情報をモデムに発生できるように、汎用モデムドライバによってモデム特定制御情報を読み取ることと、

前記呼コントローラにおいて呼制御コマンドを受けることと、

前記通信ドライバにおいてデータI/Oコマンドを受けることと、

前記呼コントローラにおいて呼制御コマンドを受けたことに応じて呼制御コマンドを前記汎用モデムドライバに発生することと、

40 前記通信ドライバにおいてデータI/Oコマンドを受けたことに応じてデータI/Oコマンドを前記汎用モデムドライバに発生することと、

前記モデムがコマンドモードの場合、モデム特定制御コマンドを汎用モデムドライバによってこのモデムに転送することと、

前記モデムがデータモードの場合、データI/Oコマンドを汎用モデムドライバを経てこのモデムに転送することとを具えることを特徴とする方法。

50 【請求項11】 請求項10に記載の方法において、前記レジストリデータ記憶装置と通信し、モデム情報フ

ファイルからモデム特定制御情報を前記レジストリデータ記憶装置にロードするクラスインストーラを設けるステップと、

前記モデムと通信し、このモデムの形式を検出するエニユメレータ検出モジュールを設けるステップと、

前記モデムが前記コンピュータシステムのバスに接続されている場合、前記エニユメレータ検出モジュールによってモデム形式を検出するステップと、

モデム形式の検出に応じて、前記クラスインストーラに対して前記モデム形式を識別するステップと、

前記クラスインストーラにおいて前記モデム形式を受けるのに応じて、前記クラスインストーラによってモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項12】 請求項11に記載の方法において、前記検出ステップが、前記モデムからモデム識別情報を読み取ることを含み、前記識別ステップが、前記モデム識別情報に基づくデバイスIDを前記クラスインストーラに報告することを含みことを特徴とする方法。

【請求項13】 請求項10に記載の方法において、前記モデムと通信し、モデム形式を検出するクラスインストーラを設けるステップと、

前記クラスインストーラによって質問の組を前記モデムに対して発生するステップと、

前記質問の組に対する前記モデムの応答を受けるステップと、

前記モデムの応答を、前記質問の組の発生された質問に依存してきれいにし、きれいにした応答を得るステップと、

前記きれいにした応答を個別デバイスIDに変換するステップと、

前記個別デバイスIDに適合するものを見つけようとするステップと、

前記個別デバイスIDに適合するものが見つかった場合、前記個別デバイスIDに対応するモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項14】 請求項10に記載の方法において、前記レジストリデータ記憶装置と通信するコンフィギュレーションマネージャと、前記汎用モデムドライバをロードするデバイスローダとを設けるステップと、

前記コンフィギュレーションマネージャによって前記レジストリデータ記憶装置を読み取り、前記デバイスローダを決定するステップと、

アプリケーションプログラムが前記制御モジュールにコマンドを発生し、前記モデムとの通信を確立する場合、前記デバイスローダによって補助記憶装置から前記汎用モデムドライバをメインメモリにロードするステップとをさらに含むことを特徴とする方法。

【請求項15】 コンピュータシステムへの導入中にモ

デムを検出する方法において、

前記コンピュータシステムの通信ポートを経て第1の質問を発生し、モデムが装着されているかどうかを決定することを具え、

モデムが応答した場合、

前記モデムにモデム識別情報を送信させるように命令する質問を発生するステップと、

前記質問に対するモデムの応答を受けるステップと、

前記質問の発生された質問に依存して前記モデムの応答をきれいにし、きれいにした応答を得るステップと、

前記きれいにした応答をデバイス識別に変換するステップとを実行することを特徴とする方法。

【請求項16】 請求項15に記載の方法において、

前記コンピュータシステムに装着されたモデムについての情報を保持するレジストリデータ記憶装置と、各々がデバイス識別に関係し、モデム特定制御情報を含む情報ファイルとを設けるステップと、

前記変換ステップから得たデバイス識別に適合するものを見つけようとするステップと、

前記デバイス識別に適合するものが見つかった場合、前記デバイス識別に対応するモデム特定制御情報を前記レジストリデータ記憶装置にロードするステップとをさらに含むことを特徴とする方法。

【請求項17】 請求項15に記載の方法において、

前記デバイス識別に適合するものが見つからない場合、汎用デバイス識別を発生するステップをさらに含むことを特徴とする方法。

【請求項18】 請求項15に記載の方法において、前記変換ステップが、

前記きれいにした応答に巡回冗長検査を行うことを特徴とする方法。

【請求項19】 請求項15に記載の方法において、巡回冗長検査表を瞬時に計算することを特徴とする方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、モデムを使用するコンピュータ通信と、さらに特にコンピュータシステムにおけるモデムインタフェースとに関するものである。

【0002】

【発明が解決しようとする課題】今日のパーソナルコンピュータにおけるアプリケーションプログラム（アプリケーション）の数の増加は、リモートデータ通信の特徴をサポートするモデムと相互に影響し合っているに違いない。モデムにアクセスするアプリケーションのいくつかの例としては、リモートネットワークアクセスや、掲示板サービスや、ファックスモデムや、電子メール（eメール）等を含む必要があるかもしれない。ワードプロセッサや表計算プログラムのようなアプリケーションさえも、例えば、モデムを制御して他のコンピュータにファイルを転送するソフトウェアを含むことができる。モ

デムへのアクセスが、より多くのアプリケーションの重要な特徴になったことにより、アプリケーション開発者は、これらのアプリケーションにおけるモデム通信をサポートする開発コードの問題に取り組まなければならないようになった。

【0003】モデム制御のタスクは、アプリケーション開発を極めて複雑にする。モデムを制御するために、アプリケーションは、コンピュータに接続された個々のブランドおよびバージョンのモデムのコマンドセットを理解する必要がある。多くのモデムは、ジョージア州ノー  
10 クロス (Norcross, Georgia) にあるヘイズマイクロコンピュータプロダクツ社 (Hayes Microcomputer Products, Inc.) によって制定されたヘイズ標準ATコマンドセット (Hayes Standard AT Command set) によるコマンドを使用する。このATコマンドセットは、コンピュータ産業において良く知られ、広く使用されているモデムを制御し、設定するための多くのコマンドを含む。しかしながら、各々のブランドまたはバージョンのモデムは、いくつかのATコマンドと新たな特徴をサポートする追加のコマンドとを含むコマンドの独自の組み合わせ  
20 を独自に理解する。この結果、モデムにおいてサポートされるコマンドセットは、実際的に異なる恐れがある。

【0004】事態をさらに複雑にすることは、The Comité Consultatif International de Télégraphie et Téléphonie (CCITT) と、The International Telecommunications Union - Telecommunications Standardization Sector (ITU-T) とによって制定された、モデムおよびファクシミリ伝送に関する多数のプロトコルが存在することである。アプリケーション開発者は、アプリケーションがモデムを正確に制御できるようにする  
30 ために、モデムプロトコルに加えてATコマンドにも精通しなければならない。

【0005】モデム技術の進歩とともに、モデムの特徴の変化と歩調を合わせてアプリケーションを制作することは、アプリケーション開発者にとって大変困難なことになった。各々の新たな特徴は、しばしば、1つまたはそれ以上の新たなコマンドによってサポートされ、これらのコマンドは、モデムのブランドによって異なるかもしれない。異なるモデムの特徴の範囲に適合するアプリケーションの制作において、モデムのサポートに向けられたアプリケーションの部分は、極めて複雑になる恐れがある。このコードの開発は、明らかにアプリケーションの中心部分の機能の開発を減じる。たとえ、開発者が、現在のモデムの特徴をサポートする完全な仕事を行ったとしても、アプリケーションにおけるモデムサポートは、将来のバージョンのモデムのより進歩した特徴をサポートしないであろう。アプリケーションが市販された後、進歩した特徴に対するサポートを容易に追加する変更はできない。

【0006】アプリケーション開発の複雑化に加えて、  
50

多くのアプリケーションにおいてモデム制御を置くことは、資源コンテンション (resource contention) 問題を発生させる。各々のアプリケーションがモデム制御のサポートを準備する場合、アプリケーションにモデム資源を割り当てることは、不可能ではないにしても困難である。これらの事情において代表的に、モデムの制御を最初に得たアプリケーションは、自発的に制御を止めるまで、制御を継続する。この結果、モデムを制御するアプリケーションは、他のアプリケーションが電話呼出し  
10 に関係することを不可能にする。この状況において、シリアルポートドライバに対するインタフェースは、せいぜいいくつかのコンテンション管理を行えるだけである。しかしながらこのコンテンション管理は、ある程度制限される。

【0007】モデムサポートソフトウェアが存在することの追加の欠点は、これらがモデムをコンピュータシステムに導入することを困難にすることである。アプリケーションを2つ以上のモデム形式の制御に対して設定できるが、代表的に、個々のモデム形式との通信に関してアプリケーションを正確に設定するためのユーザからの追加の情報が必要とする。モデム特定情報の導入を自動的に行うことができるならば、モデムの導入は、ユーザ  
20 に対してはるかに簡単になるであろう。

【0008】モデム制御システムが存在することの制限および欠点を軽減するために、本発明は、デバイスに依存しないモデムインタフェースを提供する。本発明は、このモデムに対するデバイスに依存しないインタフェースを、コンピュータシステムにおいて動作しているアプリケーションに与える手段を含む。

【0009】

【課題を解決するための手段】本発明の実施例によれば、本発明は、コンピュータオペレーティングシステムの一部として含まれるデバイスに依存しないモデムインタフェースを具える。このインタフェースは、呼制御コマンドを受ける呼制御モジュールと、アプリケーションからデータ伝送コマンドを受ける通信ドライバとを含む。アプリケーションは、呼制御コマンドおよびデータ伝送コマンドを、呼制御モジュールおよび通信ドライバに、各々、デバイスに依存しない方法で送る。汎用モデムドライバは、呼制御モジュールからの命令に応じて、デバイス特定コマンドをモデムに供給する。特定のモデムの特有の能力を理解するために、汎用モデムドライバは、レジストリ (registry) データ記憶装置からモデム特定情報を読み取る。この情報を使用すると、汎用モデムドライバは、多数のモデム形式のためのデバイス特定コマンドを使用することができる。

【0010】汎用モデムドライバは、アプリケーションに対するモデム透過性をコマンドモードおよびデータモードに分離する。モデムがコマンドモードである場合、汎用モデムドライバは、呼制御コマンドをモデムに伝送  
50

する。モデムがデータモードである場合、汎用モデムドライバは、通信ドライバからデータI/Oコマンドをモデムに伝送する。

【0011】本発明は、追加の導入の特徴を提供する。本発明のある実施例は、デバイス特定情報をレジストリデータストア中にロードするクラスインストーラ(class installer)モジュールを含む。いくつかのモデムドライバは、コンピュータシステムにおけるバスに関してロードされたエnumレータ(enumerator)検出モジュールに対して自分自身を識別させる。導入中、デバイス識別を、クラスインストーラモジュールに移し、適切なデバイス情報をロードできるようにする。

【0012】モデムが自分自身を識別させられない場合、本発明の他の実施例は、モデム形式を識別する方法を提供する。この方法は、デバイス特定情報が得られるように、モデムに対して一連の質問をすることと、次にモデムの応答をきれいにすることを含む。応答データを、デバイス特定情報をロードするのに使用されるデバイス識別に変換することができる。

【0013】本発明の個々の実施例は、コンピュータ用通信システムが存在することに関して、幾つかの利点を提供する。本発明は、アプリケーション開発者の仕事を、はるかに簡単にする。開発者は、広範囲に多様なデバイスと通信するための簡単な共通インタフェースを使用することができる。本発明のある実施例において、モデムの制御を複数のレベルに分割することができ、最上位レベルにおいて、制御および通信アプリケーションインタフェースは、デバイスに依存しないインタフェースをモデムに提供し、より下位のレベルにおいて、汎用モデムドライバは、高レベルコマンドを、特定の形式のモデム用のコマンドに変換することができ、最後により下位のレベルにおいて、ポートドライバは、モデムデータと、特定の形式のポート用のコマンドとを変換することができる。

【0014】モデム制御のタスクをアプリケーションから取り除いた後、モデム制御インタフェースは、資源コンテンションをより効果的に行うことができる。

【0015】本発明は、モデムを導入し、設定するユーザの仕事は、はるかに簡単にする。ユーザは、モデムをただ単にプラグ接続することができ、システムが、残り

の面倒をみる。

【0016】汎用モデムドライバは、異なった形式のモデム用の個々のドライバを組み込む必要性を取り除く。汎用モデムドライバとともに、モデム特定情報のみをロードする必要がある。モデム情報の組み込みを、ユーザが、どんな形式のドライバを組み込む必要があるのか、またはどうやってそれを組み込むのかを知る必要なしに、行うことができる。

【0017】

【発明の実施の形態】図1は、本発明の実施例を実行す

るコンピュータシステム20のブロック図である。コンピュータシステム20は、その基本的な要素として、コンピュータ22と、入力デバイス24と、出力デバイス26とを含む。

【0018】一般にコンピュータ22は、バス構造体32を通じて連絡する、中央処理ユニット(CPU)28と、メモリシステム30とを含む。CPU28は、計算を行う演算器(ALU)33と、データおよび命令を一時的に記憶するレジスタ34と、アプリケーションまたはオペレーティングシステムのようなコンピュータプログラムからの命令に応じてコンピュータシステム20の動作を制御する制御ユニット36とを含む。

【0019】一般にメモリシステム30は、ランダムアクセスメモリ(RAM)および読み出し専用メモリ(ROM)半導体装置のような媒体の形式の高速メインメモリ38と、フロッピーディスク、ハードディスク、テープ、CDROM等および光または磁気記録材料を使用する他の装置のような媒体の形式の2次記憶装置40とを含む。メインメモリ38は、コンピュータのオペレーティングシステムおよび現在動作しているアプリケーションプログラムのようなプログラムを記憶する。メインメモリ38は、ディスプレイデバイスによって画像を表示するためのビデオディスプレイメモリも含む。

【0020】代表的に、入力デバイス24および出力デバイス26を、バス構造体32によってコンピュータ22に接続された周辺デバイスとする。入力デバイス24を、キーボード、モデム、ポインティングデバイス、ペンまたは入力データをコンピュータに供給する他のデバイスとしてもよい。出力デバイス26を、ディスプレイデバイス、プリンタ、サウンドデバイスまたはコンピュータからの出力データを供給する他のデバイスとしてもよい。

【0021】モデム56は、入力および出力装置24、26の双方として機能することができる。出力デバイスとして、モデムは、プログラムされたCPUの制御に従って、メモリからコマンドおよびデータを受け取ることができる。入力デバイスとして、モデム56は、ファックスまたはコンピュータデータを受け、このデータをさらに処理または記憶するためにメモリに転送することができる。モデム56は、データ、ファックスまたは音声呼出し、または一回の呼出しにおけるこれらの形式の組み合わせをサポートできる。コンピュータシステムを電話機に結合すると、コンピュータは、電話番号をダイヤルすることによって、音声呼出しを始めることができる。呼出し者は、ひとたび通信が確立すると、電話機の送受話器を通じて音声通信を続ける。音声およびデータ通信を切り換えるために、モデム56は、音声通信を開始し、コンピュータ間の通信を確立することができる。モデムを通じた多くのモードの電話通信が可能であることを、当業者には理解されたい。

【0022】モデムに対してコマンドおよびデータを物理的に転送する処理は、モデムの形式に依存して変化する。様々の標準モデム構成が存在し、本技術分野において良く知られている。これらの構成は、例えば、内蔵または外付けモデム、またはノートブック型コンピュータにおいて広く普及しているPCMCIA（パーソナルコンピュータメモ리카ードインターナショナルアソシエーション（Personal Computer Memory Card International Association））カード接続を含む。“内蔵”モデムは、インダストリースタンドアーキテクチャ（Industry Standard Architecture（ISA））バス、エキスパンデッドインダストリースタンドアーキテクチャ（Expanded Industry Standard Architecture（EISA））バス、またはマイクロチャネルアーキテクチャ（Micro Channel Architecture（MCA））バスのようなバス構造体32を経てコンピュータに結合される。モデムを、PCMCIAポートを経てコンピュータのバス構造体32に結合することもできる。“外付け”モデムは、RS-232シリアルポートまたはパラレルポートを経てコンピュータのバス構造体32に結合する。追加の例としては、モデムを、PCI標準（Peripheral Component Interconnect、Peripheral Component Interconnect Special Interest Groupによって支持されるローカルバス標準）に準拠したローカルバス構造体に結合してもよい。多くの他の変形例が可能であり、当業者にはよく知られている。

【0023】図1は、いろいろな用途のあるコンピュータシステムの基本的な要素を説明するブロック図であり、本図は、コンピュータシステム20の特別なアーキテクチャを説明することを意図していないことを理解されたい。例えば、コンピュータ設計の分野において既知の種々のバス構造体を、多くの方法において、望んだようにコンピュータシステムの要素の相互接続に使用できるため、個々のバス構造体を示していない。CPU28を、別個のALU33、レジスタ34および制御ユニット36によって構成してもよいし、マイクロプロセッサにおけるように、これらのCPUの部品と一緒に集積した1つのデバイスとしてもよい。さらに、コンピュータシステムの要素の数および配置を、当技術分野において既知の方法（例えば、多数プロセッサ、クライアントサーバシステム、コンピュータネットワーキング、等）において示され、記述されているものから変えてもよい。

【0024】図2は、コンピュータシステム内の、本発明の実施例のアーキテクチャを説明するブロック図である。このアーキテクチャは、通信アプリケーションにコンピュータシステム20に導入されたモデムへのハードウェアに依存しないアクセスを与えるモデムサブシステムを示す。この図は、ワシントン州レッドモンド（Redmond, Washington）のマイクロソフト社（Microsoft Co

poration）によって開発されたWindows 95オペレーティングシステムにおけるモデムサブシステムの特別なアーキテクチャを示す。Windows 95オペレーティングシステムのアーキテクチャにおいて、DOS アプリケーションのような仮想計算機が属するリング3（50）と仮想デバイスが属するリング0（52）とを含む多数のレベルが存在する。この図は、Windows 95環境におけるモデムサブシステムの特別な例を説明するが、本発明は、この特別な実現に限定されるものではないことを理解されたい。例えば、モデムサブシステムのアーキテクチャを、オペレーティングシステムが設計されているCPUの形式に依存して変えることができる。モデムサブシステムの構造に対する多くの変形例が、本発明の範囲から逸脱することなしに可能である。

【0025】モデムサブシステムは、モデム56へのアクセスを必要とする1つまたはそれ以上の通信アプリケーション54に対するサポートを提供する。通信アプリケーション54は、リモートネットワークアクセス、ファックスアプリケーション、eメールアプリケーション、掲示板サービスアプリケーション、等を含む。これらおよび他の種々のアプリケーションは、モデムへのアクセスを必要とし、モデムを制御し、モデムを経てデータI/Oを行う。

【0026】テレフォニアアプリケーションプログラミングインタフェース（Telephony Application Programming Interface（TAPI））モジュール58（または、“呼コントローラ”）は、モデム56を制御するアプリケーションにデバイスに依存しないインタフェースを提供する。この実現において、TAPIモジュール58を、動的にリンクされたライブラリ（dynamic linked library（DLL））とする。TAPI60を経てアプリケーション54によって呼ばれると、TAPIモジュール58は、国際呼出しのダイヤル、ある形式の国内呼出しに関する待機、国内呼出しの応答、現在の呼出しの停止、等のようなモデム制御機能をサポートする。

【0027】TAPIモジュール58は、アプリケーション機能呼び出しを、デバイス特定制御を行うサービスプロバイダ62に送る。このモデムサブシステムにおいて、サービスプロバイダを、ユニモデムサービスプロバイダ（unimodem service provider）62とする。

【0028】ユニモデムサービスプロバイダ（unimdm.lsp）62は、TAPIモジュール58によってテレフォニアサービスプロバイダインタフェース64を経て呼び出された場合、モデムデバイスの制御を行うDLL実現機能でもある。これらのモデム制御機能は、ダイヤル、応答、停止等を含む。ユニモデムサービスプロバイダ62に、アプリケーションによって呼び出し、ユーザがモデムパラメータを設定するのを許可するダイアログウィンドウを表示させてもよい。

【0029】データI/Oを呼制御APIに組み込むこ

とができたとしても、このモデムサブシステムは、この目的のために分離した通信インタフェースを提供する。Windows 95オペレーションシステムにおいて、アプリケーションは、通信アプリケーションプログラミングインタフェース66 (COMM API) に対して機能呼び出しを行い、モデム56を設定し、このモデムを通じてのデータI/Oを行う。

【0030】モデムサブシステムにおいてCOMM API 66をサポートするモジュールは、通信ドライバ68および仮想通信ドライバ(VCOMM) 70を含む。通信ドライバ68は、リング3レベルモジュールであり、アプリケーションが、通信デバイスをオープンし、通信デバイスから読み出し、通信デバイスに書き込むことを許可する機能を含む。通信ドライバ68は、COMM API 66をサポートし、通信アプリケーションに装置に依存しない方法でモデムを使用することを許可する。Windows 95オペレーティングシステムにおいて、通信ドライバは、Win32 カーネルの部分である。しかしながら、これは、本発明の変形例の1つに過ぎず、通信ドライバを、当業者には既知の種々の慣例的な方法において実現することができることを理解されたい。

【0031】VCOMM 70は、リング0レベルの仮想デバイスであり、システムにおける通信資源へのすべてのアクセスを管理する。これらの“通信資源”は、1つの非同期データストリームを供給する物理的または論理的なデバイスを含む。通信資源の例としては、シリアルポート、パラレルポート、およびモデムがある。

【0032】通信ドライバ68は、VCOMMによって供給される呼機能(72)によって、VCOMMと通信する。VCOMMは、コンピュータシステムにおける通信資源に対するポートに依存しないインタフェースとして機能する。

【0033】COMM API 66は、データI/O機能、ユーザインタフェース機能、およびコンフィギュレーション(configuration)機能を含む。アプリケーションは、データI/O機能と呼出し、モデムデバイス56を通じてデータを送り、かつ受ける。ユーザインタフェース機能を経て、ユーザは、モデムの好適な動作特性を選択することができる。ダイアログボックスは、スピーカのオン/オフ、ボーレート、圧縮およびエラー訂正プロトコル、等のモデムパラメータを含むことができる。アプリケーションは、コンフィギュレーション機能と呼出し、モデム56の好適な動作特性を決定または変更する。これらのコンフィギュレーション機能に応じて、通信ドライバ68は、VCOMMを経てコンフィギュレーション機能呼び出しを汎用モデムデバイスドライバ(ユニモデムデバイスドライバまたはunimodem.vxd) 74に送る。

【0034】VCOMMを、通信ドライバ68およびユニモデムデバイスドライバ74に対するポートに依存し

ないインタフェースとする。VCOMM 70は、ポートドライバに対するアクセスを与え、通信資源におけるハードウェア動作を実行する。ポートドライバは、レジスタに対するVCOMM 70サービスをそれら自身に使用し、通信ハードウェアに対するアクセスを制御する。これらのサービスに加えて、VCOMM 70は、クライアント仮想デバイスサービスを提供し、仮想デバイスが、ポートドライバが組み込まれているどのような通信資源も使用できるようにする。特に、VCOMM 70は、通信資源のオープンおよびクローズと、通信資源の設定と、通信資源からの読み出しおよび通信資源への書き込みと、通信イベントの処理と、ポートドライバの拡張機能の呼び出しとを行うサービスを提供する。

【0035】ポートドライバは、デバイス特定コードを含み、アプリケーションが、ポートに装着された物理的または論理的な特定のデバイスと通信することを可能にする。ポートドライバの一例を、図2に示すシリアルポートドライバ(serial.vxd)とする。シリアルポートドライバは、モデムサブシステムが、シリアルポート71を経てモデム56のようなシリアル通信デバイスと通信することを可能にする。このシリアルポートドライバ76は、Windows オペレーティングシステムとともに設けられ、その動作は既知である。モデム56を、多くの異なる構成においてコンピュータに結合できることから、他のポートドライバを使用してもよいことを理解されたい。この実現において、ポートドライバを、ブートタイムにおいてまたは要求されたときにメモリ内にロードする。

【0036】ポートドライバを要求されてロードする一例として、Windows 95オペレーティングシステムにおけるプラグアンドプレイアーキテクチャ(Plug and Play Architecture)に準拠するポートドライバを、ドライバがサポートするポートをVCOMMクライアントが開こうとする場合にロードする。プラグアンドプレイアーキテクチャは、マイクロソフト社によって開発されたWindows オペレーティングシステム環境のために設計された導入システムである。このアーキテクチャの関連した部分を、図3の導入サブシステムの参照とともに詳細に記述する。

【0037】図2のモデム導入サブシステムにおいて、汎用モデムドライバは、TAPI 60からの呼制御機能およびCOMM API 66からのデータ伝送機能の両方のためのインタフェースを提供する。汎用モデムドライバは、ユニモデムサービスプロバイダ62と呼ばれるテレフォニサービスプロバイダと、ユニモデムデバイスドライバ(unimodem.vxd) 74と呼ばれるVCOMMポートドライバとを含む。ユニモデムサービスプロバイダ62は、ユニモデムデバイスドライバの保護モードインタフェースにおいて実行される機能(78)に対する呼び出しを行うことによって、ユニモデムデバイスドライバ



74と通信する。ユニモデムサービスプロバイダ62とユニモデムデバイスドライバ74との間のインタフェースを、当業者に既知の慣例的な技術を使用する他の方法において実現してもよい。

【0038】ユニモデムデバイスドライバは、モデム56に対する直接呼出しは行わないが、VCOMM70に戻って呼出しを行い、モデム56と通信する。ポートドライバに対するインタフェースとしてVCOMM70を使用することによって、ユニモデムデバイスドライバ70は、ポートに依存しなくなる。この実現において、示したポートは、シリアルポートドライバ(serial.vxd) 10 76であるが、ユニモデムデバイスドライバ74は、いかなる形式のポートに結合されたモデム56も制御できる。

【0039】ユニモデムデバイスドライバ74は、モデム56とのすべての通信を制御する。この制御を実行するために、設定およびデータI/Oに加えて呼制御は、ユニモデムデバイスドライバ74を通過する。呼制御のために、ユニモデムサービスプロバイダ62は、ユニモデムデバイスドライバを呼び出し、ダイヤル、応答、停止、等のようなモデム制御コマンドを起動する。ユニモデムサービスプロバイダ62は、これらの制御呼出しを、インタフェース78を使用するVCOMMを経てユニモデムデバイスドライバ74に渡す。これに応じて、ユニモデムデバイスドライバ74は、これらの呼制御機能を、コンピュータシステム20に結合されたモデム56の形式に適合するフォーマットに変換する。VCOMM70に戻る呼びだし(80)を行うことによって、ユニモデムデバイスドライバ74は、モデムコマンドをシリアルポートドライバ76にVCOMM70を経てポート 30 に依存しない方法において転送する。

【0040】COMM APIからのデータ転送機能に関して、ユニモデムデバイスドライバ74は、データI/O呼出しを監視する。データI/O呼出しは、アプリケーション54がCOMM APIを呼出したときに発生する。通信ドライバ68は、VCOMM70を経てこれらの呼出しをユニモデムデバイスドライバ74に渡す。ユニモデムデバイスドライバ74は、VCOMM70を経てこれらのコマンドを、モデム56が装着されたポートドライバ76に渡す。

【0041】ユニモデムデバイスドライバ74は、モデム56の透過性をコマンドモードとデータモードとに分離する。モデム56がコマンドモードにある場合、ユニモデムデバイスドライバ74は、ポートドライバ76を経てモデム制御コマンドをモデム56に渡す。反対に、モデム56がデータモードにある場合、ユニモデムデバイスドライバ74は、ポートドライバ76を経てデータI/Oコマンドをモデム56に渡す。ユニモデムデバイスドライバ74は、サービスプロバイダ62およびアプリケーション54に関連して、透過性をコマンドおよび 50

データモード間で切り換える。"透過"制御は、ユニモデムサービスプロバイダ62が、データモード中にモデム56にデータが送られているのを知らず、アプリケーション54が、コマンドモード中にモデム56に呼制御コマンドが送られているのを知らないことを意味する。

【0042】モデムがコマンドモードの場合、設定および監視に関するCOMM API機能が処理される。例えば、アプリケーション56が、データの10バイトを受けた場合そのことを知らせるためにCOMM API 66において呼出しを行った場合、コマンドモードにおいてモデム56から受けたバイトは、計数されない。モデム56がデータモードに切り換わり、10バイトが受けられるとすぐに、アプリケーション54は通知される。

【0043】モデムがコマンドモードの場合、ユニモデムデバイスドライバは、アプリケーションからのデータI/Oコマンドを処理しない。しかしながら、ユニモデムデバイスドライバは、COMM APIからのモデム設定呼出しと、呼制御に関するいくつかの拡張機能を処理することができる。これらの拡張機能は、初期化、ダイヤル、および傾聴のような機能を含む。初期化拡張は、モデムを適正設定に初期化する。ユニモデムデバイスドライバは、TAPIモジュールに通知することによって初期化が完了した場合、報告する。ダイヤル拡張は、ユニモデムデバイスドライバに番号をダイヤルすることを要求する。ユニモデムデバイスドライバは、モデムが接続した場合および接続に失敗した場合に報告する。傾聴拡張は、ユニモデムデバイスドライバに到来した呼出しに答えることを要求する。

【0044】汎用モデムドライバ74は、多様なモデム形式をサポートすることができる。個々のモデム56をサポートするために、デバイス特定情報をレジストリにロードする導入サブシステムが存在する。一度この情報がロードされると、汎用モデムドライバは、デバイス特定コマンドを使用してモデムと通信できるようになる。この導入サブシステムについて、図3の参照とともに以下に記述する。

【0045】図3は、本発明の実施例による導入サブシステムアーキテクチャを説明するブロック図である。導入サブシステムは、コンフィギュレーションマネージャ90、エニユメレータ92、クラスインストーラ94、デバイスローダ、およびレジストリデータ記憶96を含む。本サブシステムにおいては、上述した仮想デバイスVCOMM70を、デバイスローダとする。

【0046】コンフィギュレーションマネージャモジュール90を使用し、モデムハードウェアのような物理的デバイスと、通信ポートのような論理的装置とを含む本システムにおける通信資源の管理を行う。Windowsオペレーティングシステムにおいて、コンフィギュレーションマネージャ90は、リング0に属する仮想デバイスで

ある。特定のローディングの位置および方法が本発明に不足しているとしても、この実現におけるコンフィギュレーションマネージャ90は、システム起動時にオペレーティングシステムによってロードされる。コンピュータシステム20をオンにした場合、CPUは、その起動ルーチンを実行する。この段階の間に、CPUは、コンフィギュレーションマネージャ90を含むオペレーティングシステムをロードする。コンフィギュレーションマネージャ90は、レジストリ96内の設定情報を含むデータ構造体の情報を読み取ることによって、システム構成について学ぶ。コンフィギュレーションマネージャ90は、エニユメレータをロードし、コンピュータシステム20に結合されたモデムデバイスの識別を助ける。コンフィギュレーションマネージャ90は、資源の競合の検査と、システムにおけるデバイスについての情報の、この場合にはVCOMM70であるデバイスローダへの報告も行う。コンフィギュレーションマネージャ90は、新たなデバイスがシステム構成に追加されたことを決定すると、クラスインストーラ94を呼出し、新たなデバイスについての情報をレジストリ96にロードする。

【0047】エニユメレータ92は、コンピュータシステムに装着されたデバイスを識別するのに使用される検出モジュールである。Windows 95オペレーティングシステムにおいて、エニユメレータ92は、リング0に属する仮想デバイスである。起動時に、コンフィギュレーションマネージャ90は、各バスに対して1つのエニユメレータをコンピュータシステム20にロードする。コンピュータシステム構成に依存して、コンフィギュレーションマネージャは、例えば、ISA、PCMCIA、PCI (Peripheral Component Interconnect、Peripheral Component Interconnect Special Interest Groupによって支持されるローカルバス標準)、シリアルCOM、等の、1つまたはそれ以上のエニユメレータをロードすることができる。エニユメレータ92を起動時に使用し、新たに導入された装置を検出する。PCMCIA互換デバイスのようないくつかのモデムは、起動後にロードできることから、エニユメレータ92を、ランタイムにおいて装着されたモデムデバイスの検出にも使用する。

【0048】ランタイム前かランタイム中に装着された、新たに導入されたモデムは、デバイス識別(ID)によってそれ自身を識別できる。デバイスIDを、モデム56の形式を識別する単一番号または文字列としてもよい。モデム56が自分自身を識別できる、多くの他の方法が存在する。例えば、モデムは、接続されていることを単に識別し、特別なデバイスIDを与えない。このようにした場合、エニユメレータを、デバイス特定情報を読み取り、制御および設定情報がコンピュータシステムに格納されているモデムデバイスを記述するデータを

有するこの情報に適合させようとするように設計することができる。

【0049】いくつかの場合において、導入サブシステムは、モデムに質問してモデム特定情報を発生させ、この情報からデバイスIDを得る必要があるかもしれない。この追加のモデム検出について、クラスインストーラ94と関連して以下に記述する。エニユメレータ92は、デバイスIDを成功裏に決定した場合、このデバイスIDをコンフィギュレーションマネージャ90に報告する。つぎにコンフィギュレーションマネージャ90は、このデバイスIDを有するクラスインストーラを呼出し、その結果、新たなデバイスのデバイス特定情報をレジストリ96にロードすることができる。

【0050】レジストリ96は、コンピュータシステム20についての情報を保持するデータ構造体である。モデムを導入する状況において、レジストリ96は、コンピュータシステム20に結合された1つまたはそれ以上のモデムについての情報を保持する。このモデム情報を見失わないようにするために、レジストリ96は、ハードウェア接続情報とモデム制御および構成情報とを示す項目の列を含む。モデムがコンピュータシステムに導入されると、これらの項目は、レジストリ96に追加される。モデム情報を、ユーザがモデム情報を選択および/または入力することを可能にするユーザインタフェースの支援によってレジストリ96に追加することができる。モデム情報を、クラスインストーラ94によってレジストリに追加することもできる。

【0051】本実施例において、導入サブシステムは、デバイスがWindows 95オペレーティングシステムのプラグアンドプレイアーキテクチャに準拠しているかどうかによって異なって動作する。非プラグアンドプレイデバイスの場合、ユーザは、導入過程の間、レジストリにロードすべき情報を指定する。これは、レジストリにロードすべきファイルにおけるユーザ指定情報を含んでもよい。ユーザは、ユーザインタフェースにおける制御パネルを通じて、情報を選択および/または入力することができる。ユーザがモデム情報を対話式に選択および入力できるのに加えて、プラグアンドプレイアーキテクチャは、ユーザの介入なしにモデムを導入することができる。プラグアンドプレイデバイスの場合、エニユメレータ92は、このデバイスがコンピュータシステムにおけるバスに結合されているかどうかを検出する。プラグアンドプレイデバイスを、コンピュータシステムをオンにする前にバスに装着することができ、いくつかの場合において、ランタイムにおいてバスに装着することができる。いずれの場合においても、エニユメレータ92は、デバイス特定情報をバスから読み出し、デバイスIDをコンフィギュレーションマネージャ90に報告する。コンフィギュレーションマネージャ90が、モデム56が新たなものであると決定した場合、クラスインストーラ

94を呼出し、デバイス特定情報をINF（情報）ファイル98からレジストリにロードする。

【0052】INFファイル98は、1つまたはそれ以上のモデムに関する特定および制御コマンドを記述する1つまたはそれ以上のスクリプトファイルを含む。INFファイルを、コンピュータシステムの補助記憶装置に記録し、必要な場合にアクセスすることができる。モデムスクリプトファイルは、モデムの機能と、モデムの制御に必要なATコマンドとについての記述を含む。クラスインストーラが、モデムに関するINFファイル98からの情報を追加した後、ユニモデムデバイスドライバ74は、レジストリ96におけるこの情報を、モデムの適切な制御に使用することができる。

【0053】クラスインストーラ94は、動的にリンクされたライブラリであり、モデムを識別し、コンピュータシステム20に導入することに使用することができる。デバイスIDを受けるのに応じて、クラスインストーラ94は、スクリプトファイルから固有情報をレジストリ96にロードする。クラスインストーラは、ユーザの要求に応じて、モデムスクリプトをレジストリにロードすることができる。代わりに、クラスインストーラ94は、ユーザの介在なしに、デバイス特定情報をレジストリにロードすることもできる。エニユメレータ92がモデム56を認識した場合、コンフィギュレーションマネージャは、デバイスIDをクラスインストーラ94に渡す。次に、クラスインストーラは、固有モデムスクリプトをレジストリに自動的にロードする。

【0054】エニユメレータがモデムの確認を決定できない場合、クラスインストーラを、モデムの形式の検出に使用することができる。クラスインストーラ94は、まだ未確認のモデムにコマンド列を発生することによって、モデムに質問する。質問に応じて、モデムは、自分自身についての情報を伝送することによって応答する。モデムを確認するためだけには、多くの情報は利用しないことから、モデムの応答をきれいにし、32ビット巡回冗長検査を最後に行い、32ビット数を発生させる。次にクラスインストーラ94は、この数を使用し、INFファイル98におけるモデムスクリプトに対応する適合するデバイスIDを見つける。

【0055】図4および図5は、モデムの形式を検出する導入サブシステムにおいて実行されるステップを示す。検出方法は、モデムの存在に関して装着されているモデムを有する可能性がある各通信ポートの検査を含む。ポートにおいてモデムが存在しているかどうかを検\*

#### 質問リスト:

"ATI0\r",	"ATI1\r",	"ATI2\r",	"ATI3\r",
"ATI4\r",	"ATI5\r",	"ATI6\r",	"ATI7\r",
"ATI8\r",	"ATI9\r",	"ATI10\r",	"AT%V\r"

【0060】表1の各々のコマンドは、モデムに命令し、そのメモリに自分自身についての情報に関して質問

\*査するために、クラスインストーラ94は、最初にVCOMM70を呼出し、このポートを開く。次にVCOMM70は、このポートに関するポートドライバ76をロードし、クラスインストーラ94とこのポートとの間の通信を確立する。VCOMM70を使用して、ポートを開くステップと通信を確立するステップとは、マイクロソフト社から入手できるVCOMM APIに記載されている。

【0056】通信が一度確立すると、クラスインストーラ94は、質問を発生し、モデムが接続されている場合、これを決定する(100, 102)。この質問は、ATE0Q0V1であり、ATコマンドセットからの標準的なコマンド列である。逐語的にこの質問は、エコーオフ(E0)、応答する(Q0)、詳しい形式で応答する(V1)を意味する。

【0057】クラスインストーラ94は、標準ボーレートの各々において、初期質問を4回発生する(104, 106)。これらのレートは、9600、2400、1200、および300を含む。あるボーレートにおいてモデムが検出された場合、クラスインストーラ94は、次のステップに進む。もしそうでなければ、クラスインストーラ94は、次の通信ポートに移り、初期質問を繰り返す。

【0058】モデムが検出された場合、クラスインストーラは、デバイスIDを構築しようとする。図4および図5に示す次のステップは、巡回冗長検査プログラムを初期化するステップである(108)。このステップにおいて特別に行われる初期化は、決定的なものではない。しかしながら、デバイスIDを発生するCRC計算の前に行うべきである。この実施例において、CRC表を瞬時に構築する。この形式のCRCは、CRC表の静的なコピーを保持する必要性がなくなるため、性能が向上する。この形式のCRCプログラムの代わりに、種々のCRCプログラムを使用することができるが、この形式のCRCプログラムが好適である。CRC表は、瞬時に形成されることから、CRCプログラムは、この処理の初期に初期化される。CRC表を構築する次のステップを、次のステップ110として示すが、ステップ110は、実際には瞬時に生じることを理解されたい。

【0059】図4および図5における次の2つのステップ112および114は、質問リストにおける質問を発生する処理を表す。表1は、リスト上の質問である。

#### 【表1】

させ、その情報を返させる。

【0061】質問を発生した後、クラスインストーラ

は、応答を読み取る(116)。次にこの応答を、質問の形式に従ってきれいにする。質問がATIOの場合(118)、第1ラインの第1ワードのみを使用する(120)。

【0062】質問がATI4であり、応答文字列が、ヘイズATコマンドセットにおいて示されるI4標準に適合する場合、第1ライン全体を使用する(124)。 \*

ベイルリスト:

"CONNECT", "RING", "NO CARRIER", "NO DIALTONE",  
"BUSY", "NO ANSWER", "="

【0065】文字列が、以下の表3に示す包含リストにあり、数字に囲まれていない場合(130)、文字列全※

包含リスト:

"300",			
"1200",			
"2400",			"2,400",
"9600",	"96",	"9.6",	"9,600",
"12000",	"120",	"12.0",	"12,000",
"14499",	"144",	"14.4",	"14,400",
"16800",	"168",	"16.8",	"16,800",
"19200",	"192",	"19.2",	"19,200",
"21600",	"216",	"21.6",	"21,600",
"24000",	"240",	"24.0",	"24,000",
"26400",	"264",	"26.4",	"26,400",
"28800",	"288",	"28.8",	"28,800",
"31200",	"312",	"31.2",	"31,200",
"33600",	"336",	"33.6",	"33,600",
"36000",	"360",	"36.0",	"36,000",
"38400",	"384",	"38.4",	"38,400",
"9624",	"32bis",	"42bis",	"V32",
"V.32",	"V.FC",	"FAST",	"FAX",
"DATA",	"VOICE"		

【0066】文字列が、以下の表4に示す排除リストに 30★【表4】  
ある場合、この文字列をスキップする(136)。 ★

排除リスト:

"JAN", "FEB", "MAR", "APR", "MAY", "JUN",  
"JUL", "AUG", "SEP", "OCT", "NOV", "DEC"

【0067】上記のステップのいずれもが、第1ラインの文字列に対して満足されない場合、この文字列を次のようにきれいにする。

- 1) すべての数字を除去する(138)。
- 2) 非6文字に隣接していないすべての6文字を除去 40する(140)。
- 3) それら自身によってすべての文字を除去する(142)。
- 4) すべてのピリオド、カンマ、および間隔を除去する(144)。

【0068】表1のリスト上のすべての質問を発生した後、巡回冗長検査を、きれいにした応答における各々の文字または数字に関して計算する(146)。CRCの結果は、32ビット数である。次に32ビットCRCを、"UNIMODEMxxxxxxx"の形式のデバイスIDを表す 50

\*【0063】他のすべての質問に関して、第1ラインを以下のようにきれいにする。

【0064】文字列が、以下の表2に示すベイル(bail)リストにある場合(126)、ベイルリスト中の文字列に格納されている部分のみを使用する(128)。

【表2】

※体を使用する(132)。

【表3】

文字列に変換する(148)。

【0069】次にクラスインストーラ94は、デバイス情報がINFファイル98に含まれるデバイスIDのリストを探索する必要がある。適合するものが見つかった場合(150)、クラスインストーラは、適切なスク립トをレジストリ96にロードすることができる。

【0070】しかしながら適合するものが存在しない場合、汎用IDを使用する(152)。汎用IDを、モデムサブシステムがモデムが接続されていることを知っているが、モデムを明確に識別できない、および/または適合するデバイスIDを見つけれない場合に使用する。汎用IDは、INFファイルにおける汎用モデム制御情報に関連する。この汎用モデム制御情報は、広く多種のモデムに大部分適合すると思われるコマンドを基礎とする。汎用制御情報を、接続された特定のモデムに対

して作り変えることができる。さらに、大体のDCエレートを、AT I O 質問に対する応答に基づいて決定することができる。

【0071】モデムサブシステムおよび導入サブシステムの双方のアーキテクチャおよび動作を記述したので、例とともに実施例の動作を説明するのに好適になった。

【0072】この処理の第1段階は、モデムのコンピュータシステムへの導入である。この導入段階は、モデムを認識することと、次にモデム特定情報をレジストリ96にロードすることを含む。処理の第2段階は、モデムに依存しないインタフェースを使用し、モデムとのデータI/Oを制御しかつ行うことを含む。次の例は、PCMCIA互換モデムデバイスに関するものである。モデムの形式、すなわち、外付けであるか、内蔵であるかと、含まれる物理的および論理的通信ポートの形式とに依存して、他の導入構成が可能であることを理解された。

【0073】代表的なPCMCIAデバイスに関して、ユーザが、コンピュータシステム20におけるPCMCIAバスに結合されたPCMCIAカードスロットにPCMCIAカードを挿入したときに、導入は開始する。ユーザは、コンピュータの起動前か、ランタイム中に、デバイスを挿入してもよい。

【0074】ユーザが、起動前にPCMCIAを導入する場合、起動時にバスにロードされたエニユメレータ92が、デバイスを検出し、そのデバイスIDをコンフィギュレーションマネージャ90に報告しようとする。PCMCIAデバイスの場合、PCMCIAコネクタに装着されたPCMCIAデバイスを検出し、報告するエニユメレータ92が存在する。同様に、ユーザが、ランタイムにおいてデバイスを挿入する場合、エニユメレータ92は、同様の検出および報告ステップを行う。エニユメレータ92が、デバイスを認識できた場合、デバイスIDをコンフィギュレーションマネージャに渡す。

【0075】次に、コンフィギュレーションマネージャ90は、デバイスIDを読み取り、モデムがシステムにおける新たなデバイスであることを認識する。コンフィギュレーションマネージャは、デバイスのリストを格納したレジストリ96を検査することによって、モデムが新たなデバイスかどうかを決定する。

【0076】モデムが、エニユメレータ92に対して自分自身を認識させるようにプログラムされていない場合、クラスインストーラ94は、デバイスIDを発生しようとする。コンフィギュレーションマネージャ90は、クラスインストーラ94を呼出し、詳細に上述した検出プログラムを実行する。

【0077】一度、デバイスIDが決定すると、コンフィギュレーションマネージャ90は、このデバイスIDを有するクラスインストーラ94を呼び出す。クラスインストーラ94は、このデバイスIDに基づく適切なI

NFファイル98を見つけ、INFファイル情報をレジストリ96にコピーする。さらにクラスインストーラ94は、PCMCIAモデムを、新たなラインデバイスとしてTAPIモジュール58に報告する。

【0078】レジストリ96を更新した後、クラスインストーラ94は、新たなモデム56が導入されたことを、コンフィギュレーションマネージャ90に報告する。次に、コンフィギュレーションマネージャ90は、レジストリ96を読み取り、デバイスローダを決定する。この実現において、VCOMM70はデバイスローダであり、レジストリ96の入口として認識される（例えば、デバイスローダ=vcomm.vxd）。上述したように、デバイスローダとしてVCOMM70は、適切なドライバのローディングを行い、モデムとの通信を確立する。

【0079】コンフィギュレーションマネージャ90は、モデムデバイスと、それに関連する通信資源とを認識するポインタを、デバイスローダ（VCOMM）70に与える。明確に、コンフィギュレーションマネージャ90は、ユニモデムデバイスドライバ74およびシリアルポートドライバ70のような、モデムに使用される1つまたはそれ以上のドライバについて、VCOMM70に知らせる。この時点で、モデム56は、導入され、アプリケーションによってアクセスされる準備ができる。

【0080】アプリケーション54が、モデム56との通信を確立したい場合、TAPIモジュール58が、ラインデバイスに質問することによって開始する。TAPIモジュール58は、モデムの認識を、モデムを認識する番号に戻す。次にアプリケーション54は、“オープン”にすべきモデムのモデム認識を指定するTAPIモジュール58に対する呼出しを行うことによって、モデム56との通信を開く。

【0081】アプリケーション54からのこの呼出しに応じて、TAPIモジュール68は、ユニモデムサービスプロバイダ62を呼出し、モデムデバイスを開く。次にユニモデムサービスプロバイダ62は、VCOMM70を呼出し、ポートを開く。

【0082】VCOMM70が、その名前によってモデムを開く呼出しを受けた場合、ユニモデムデバイスドライバ74をロードする。VCOMM70は、デバイスの名前がレジストリ96におけるデバイスドライバと関係していることから、ユニモデムデバイスドライバ74をロードすることを知り、デバイスドライバは、ユニモデムデバイスドライバとして認識される。

【0083】次に、ユニモデムデバイスドライバ74は、モデムと関連するポートを開くことを要求する。これに応じて、VCOMM70は、適切なポートドライバ（例えば、シリアルポートドライバ、serial.vxd）をロードする。

【0084】この時点から、アプリケーションは、TA

P I モジュール 58 を経て制御呼出しを行い、モデム 56 を制御する（すなわち、番号のダイヤル、国内呼出しの待機、等）。T A P I モジュール 58 は、これらの制御呼出しをユニモデムサービスプロバイダ 62 に渡し、このユニモデムサービスプロバイダ 62 は、これらの呼出しを、ユニモデムデバイスドライバ 74 のより低いレベルの呼出しに変換する。レジストリ 96 における情報を使用して、ユニモデムデバイスドライバ 74 は、モデム 56 を呼出しに対して準備するために適切に初期化する。

【0085】この時点で、アプリケーション 54 は、T A P I モジュール 58 を呼出し、モデムを介した電話接続を制御することができる。一度、他のモデムとの接続が確立すると、T A P I モジュール 58 は、この接続についてアプリケーションに通知する。

【0086】本発明を、特定の実施例の参照とともに記述したが、本発明の範囲から逸脱することなしに、多くの変形例が可能であることを理解されたい。例えば、T A P I モジュールおよび通信ドライバの構成を変え、依然としてアプリケーションに対して同様のインタフェースを与えることができる。システムアーキテクチャの多くの特徴は、Windows 95 システムの特定のアーキテクチャに基づくものであるが、この特定のアーキテクチャを変更することができることを理解されたい。例えば、他のシステムは、Windows オペレーティングシステムに関して既知の分離した“リング 3”および“リング 0”を持たなくてもよい。ユニモデムサービスプロバイダおよびユニモデムデバイスドライバを、いくつかのシステムにおいて、結合してもよい。より低いレベルにおいて、VCOMM のようなポートに依存しないインタフェースが存在することは必要なく、同様にユニモデムデバイスドライバを、VCOMM ポートドライバとする必要はない。エミュレータ、ユニモデムデバイスドライバ、およびクラスインストーラによって提供される機能を、異なるモジュールにおいて提供することができ、システムにおいて種々の方法においてロードすることができる。要するに、多くの変形例が可能である。

【図面の簡単な説明】

【図 1】本発明の実施例を実現するコンピュータシステ

ムのブロック図である。

【図 2】コンピュータシステムにおける本発明の実施例のアーキテクチャを説明するブロック図である。

【図 3】本発明の実施例の 1 つの特徴によるインストーラアーキテクチャを説明するブロック図である。

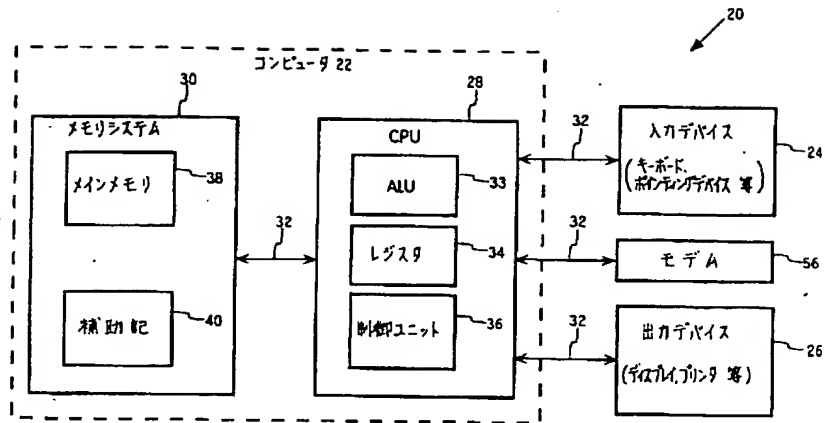
【図 4】モデムの形式を検出する導入サブシステムにおいて実行されるステップを説明するフローチャートを図 5 とともに構成する一部である。

【図 5】モデムの形式を検出する導入サブシステムにおいて実行されるステップを説明するフローチャートを図 4 とともに構成する一部である。

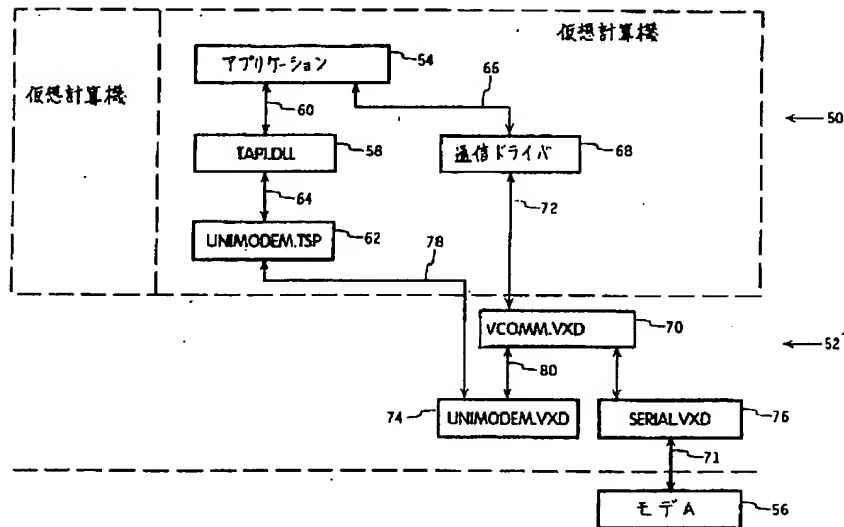
【符号の説明】

- 20 コンピュータシステム
- 22 コンピュータ
- 24 入力デバイス
- 26 出力デバイス
- 28 中央処理ユニット
- 30 メモリシステム
- 32 バス構造体
- 33 演算器
- 34 レジスタ
- 36 制御ユニット
- 38 メインメモリ
- 40 2 次記憶装置
- 50 リング 3
- 52 リング 0
- 54 通信アプリケーション
- 56 モデム
- 58 T A P I モジュール
- 60 T A P I
- 62 ユニモデムサービスプロバイダ
- 64 テレフォニサービスプロバイダインタフェース
- 66 通信アプリケーションプログラミングインタフェース
- 68 通信ドライバ
- 70 仮想通信ドライバ
- 71 シリアルポート
- 74 汎用モデムデバイスドライバ
- 76 シリアルポートドライバ

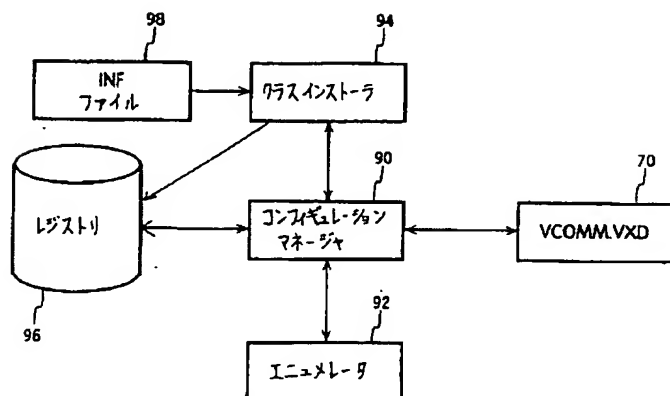
【図 1】



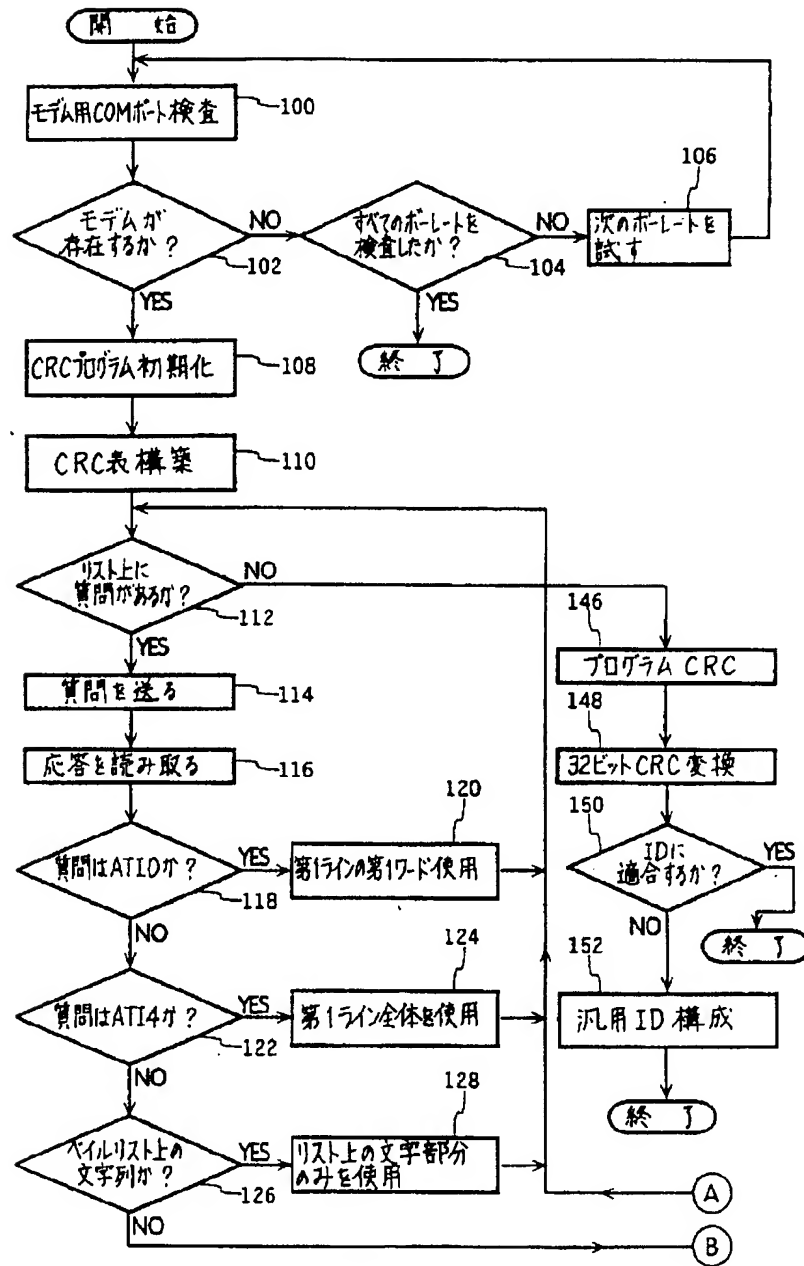
【図 2】



【図 3】

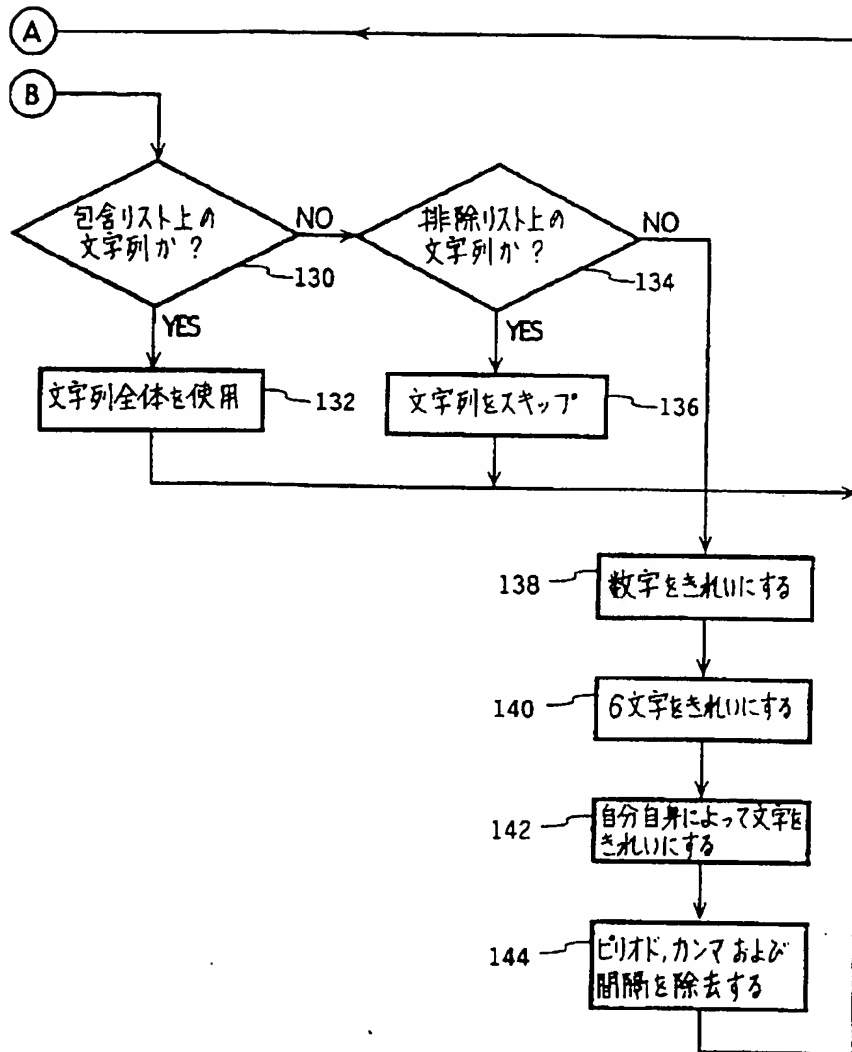


【図 4】





【図5】



フロントページの続き

(72)発明者 ロバート ジェイ ウィリアムス  
 アメリカ合衆国 ワシントン州 98033  
 カーkland ワンハンドレッドトゥエン  
 ティエイ ス アヴェニュー エヌイー  
 10020

(72)発明者 クリストファー ビー ガブト  
 アメリカ合衆国 ワシントン州 98052  
 レドモンド ウェスト レイク サマリッ  
 シュ パークウェイ エヌイー 4250 シ  
 ー 2009

(72)発明者 キース エイ リーブル  
 アメリカ合衆国 ワシントン州 98053  
 レドモンド エヌイー セブンティーン ス  
 コート 21802

**THIS PAGE BLANK (USPIC,**